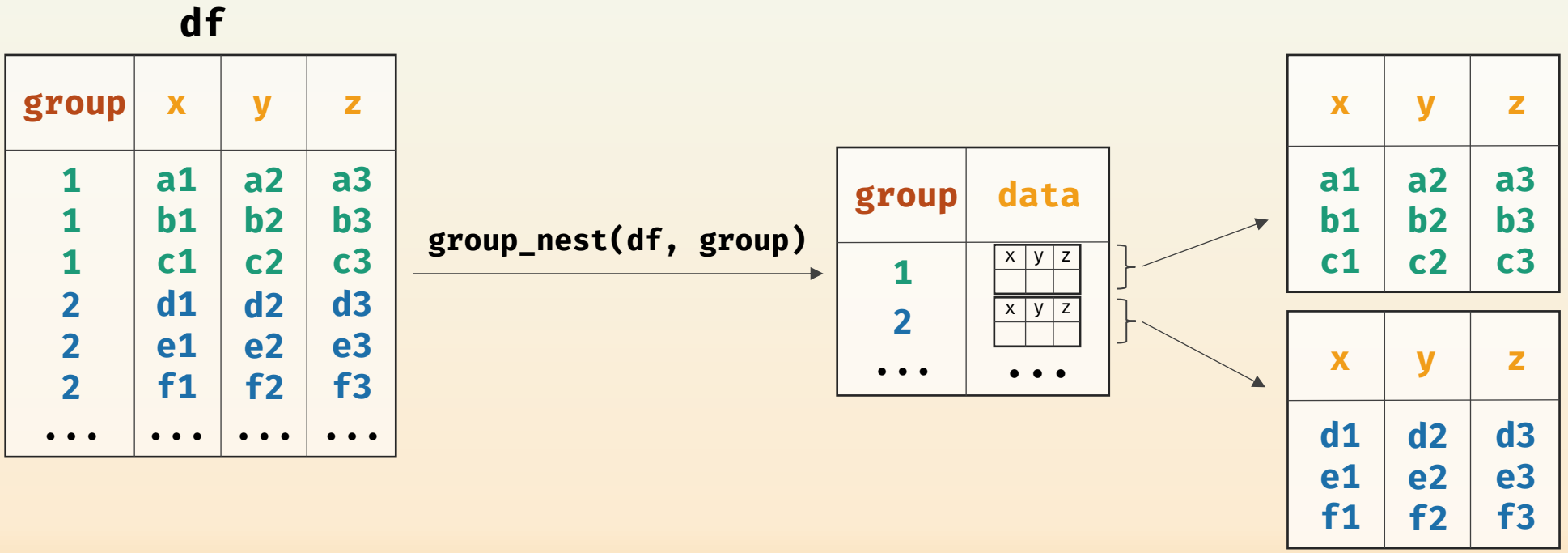
rstudio::conf(2020L)

List-columns in data.table

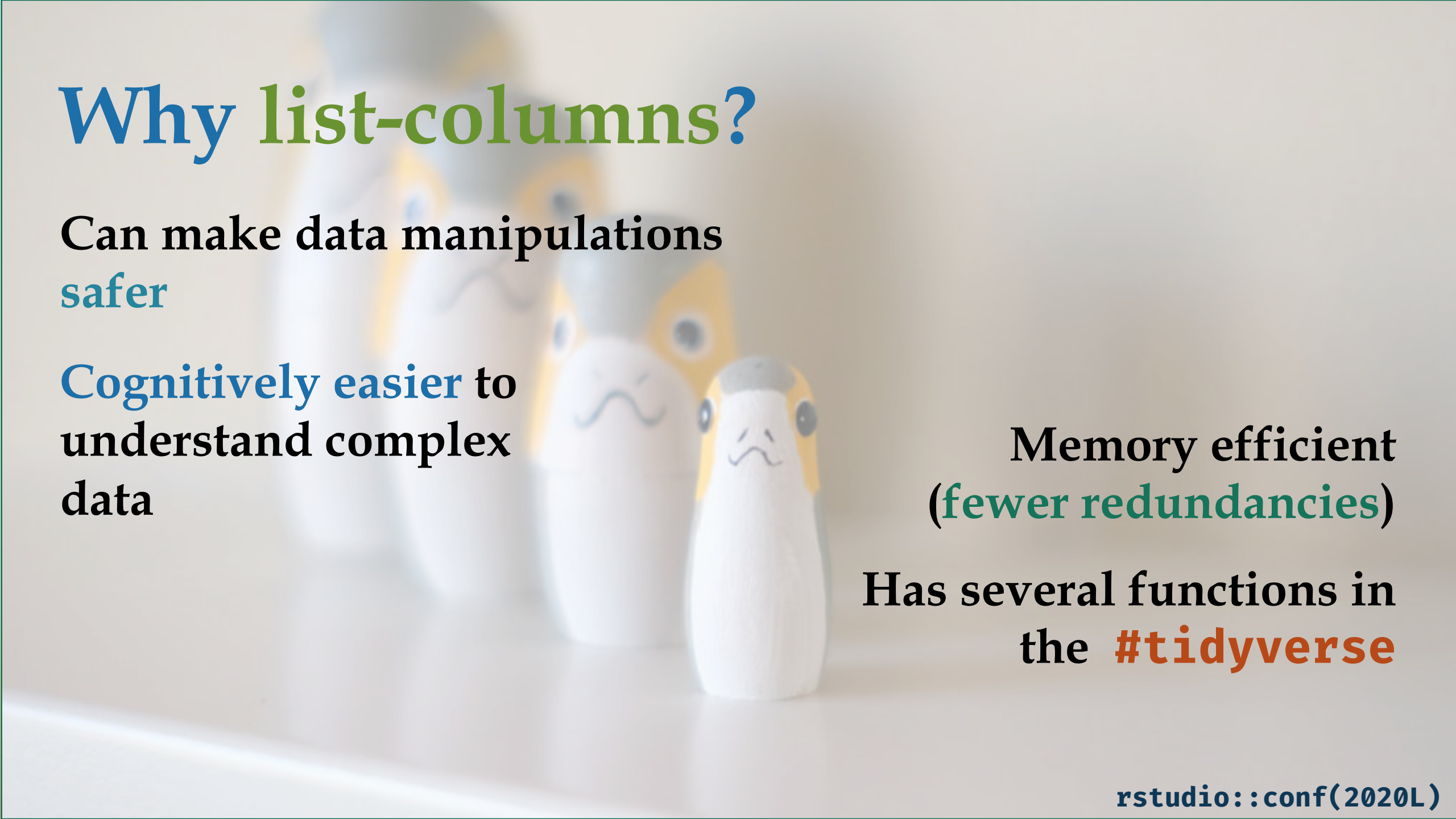
Tyson S. Barrett

Artwork from @juliesquid for @openscapes
(illustrated by @allison_horst)

What are list-columns?



Why list-columns?



Can make data manipulations safer

Cognitively easier to understand complex data

Memory efficient (fewer redundancies)

Has several functions in the `#tidyverse`

Why `data.table`?

- Concise syntax
- Fast speed
- Memory efficient
- Careful API lifecycle management
- Community
- Feature rich

*According to the GitHub README

Why data.table?

Concise syntax

Fast speed

Memory efficient

Careful API lifecycle management

Community

Feature rich

*According to the GitHub README

Why data.table?

Concise syntax

Fast speed

Memory efficient

Careful API lifecycle management*

Feature rich

```
dt[i, j, by]
```

"filter"

"mutate"

"group_by"

*According to the GitHub README

Why `data.table`?

Concise syntax

Fast speed

Memory efficient

Careful API lifecycle m

Community

Feature rich

Responding to why `data.table` is so fast, Hadley Wickham:

"I think it's a relentless focus on performance across the entire package."

https://twitter.com/hadleywickham/status/1153850194892640256

*According to the GitHub README

Why data.table?

Concise syntax

Fast speed

Memory efficient

Careful API lifecycle

Community

Feature rich

https://h2oai.github.io/db-benchmark/

Summing by group

Input table: 1,000,000,000 rows x 9 columns (50 GB)

data.table	1.12.9	2019-12-12	1339s
(py)datatable	0.10.0a0	2019-11-24	2682s
dplyr	0.8.3	2019-12-05	timeout
pandas	0.25.3	2019-12-07	out of memory
spark	2.4.4	2019-12-05	not yet implemented
dask	2.9.0	2019-12-07	timeout
DataFrames.jl	0.20.0	2019-12-09	out of memory
ClickHouse	19.16.2.2	2019-12-09	CH server crash
cuDF	0.10.0	2019-12-05	out of memory
Modin		see README	pending

*According to the GitHub README

Let's Team Up

The Grammar

Nest making a list-column of data tables or tibbles

Hoist unnesting list-columns that have vectors in them

Unnest unnesting list-columns that have data tables or tibbles in them

A Brief Example!

Data from:
- Statistica
- dplyr::starwars

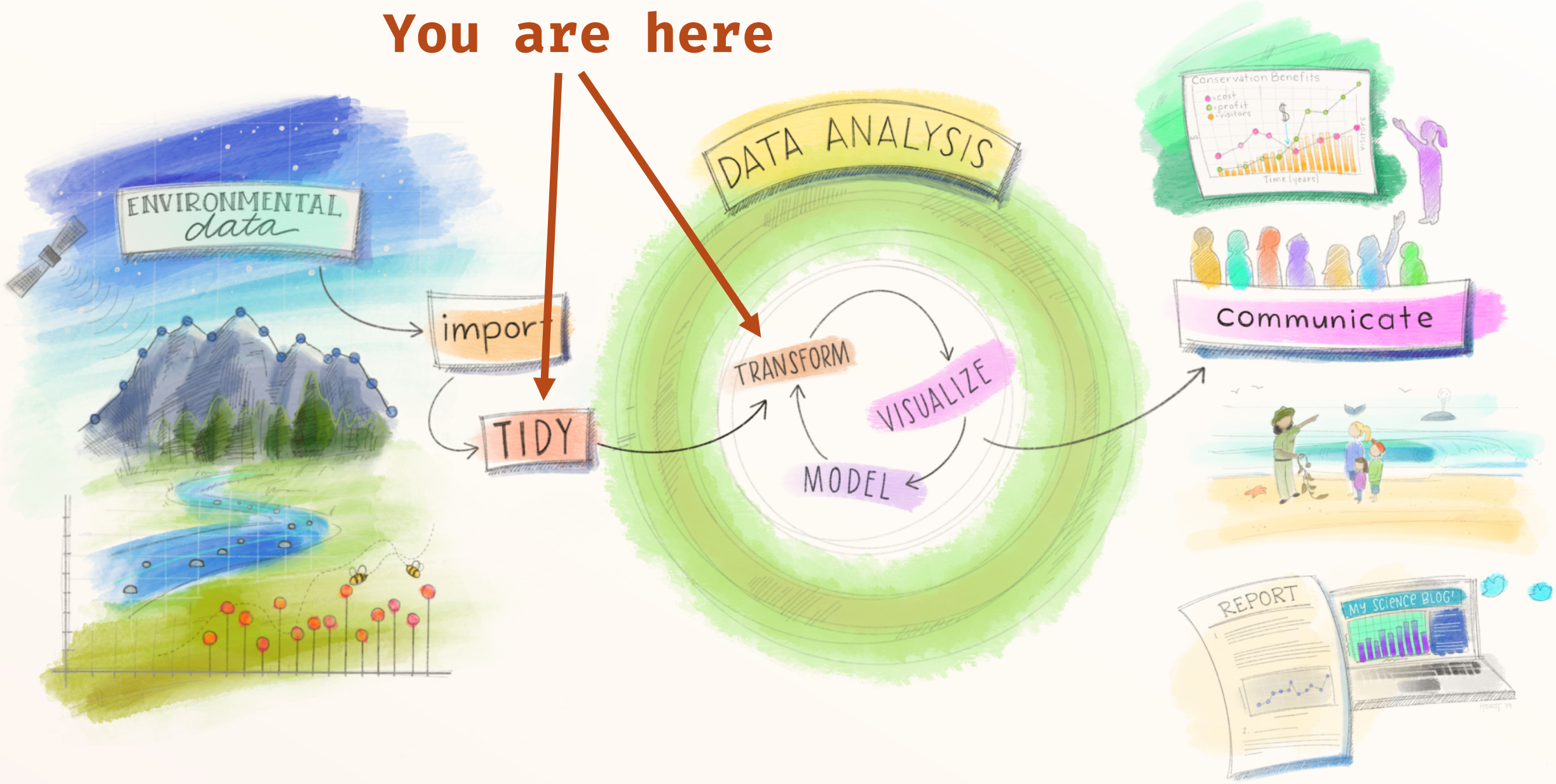
```
library(tidyverse)
library(data.table)
library(tidyfast)
```

tidyfast packages up the
functions that use data.table

The Data Sets

```
##     name                 species homeworld films      ...
##     <chr>                <chr>   <chr>     <list>     
## 1  Luke Skywalker       Human   Tatooine  <chr [5]>  ...
## 2  C-3PO                Droid   Tatooine  <chr [6]>  ...
## 3  R2-D2                Droid   Naboo     <chr [7]>  ...
## 4  Darth Vader          Human   Tatooine  <chr [4]>  ...
## ...
```

```
##    films                      revenue
##    <char>                       <num>
## 1: A New Hope                   775.4
## 2: The Empire Strikes Back      538.4
## 3: Return of the Jedi           475.1
## 4: The Phantom Menace          1027.0
...
```


You are here
ENVIRONMENTAL data
import
TIDY
DATA ANALYSIS
TRANSFORM
VISUALIZE
MODEL
Conservation Benefits
cost
profit
visitors
Time (years)
Communicate
REPORT
My Science Blog!

First, let's hoist

The variable `films` is currently a list-column of character vectors
We can hoist that column so we can use each film

The data with info on characters

```
films <- dt_hoist(dplyr::starwars,
                  films)
```

The list-column

```
##                name                                films    ...
##               <char>                               <char>    ...
##  1: Luke Skywalker     Revenge of the Sith                   ...
##  2: Luke Skywalker      Return of the Jedi                   ...
##  3: Luke Skywalker The Empire Strikes Back                   ...
##  ...
```

Then, let's nest by film

Creates a variable called data with all the data associated with each film

The data table

```
nested <- dt_nest(films, films)
nested[]
```

The list-column inside of films

The Nested Data

```
##     films                          data
##     <char>                         <list>
## 1: A New Hope                      <data.table>
## 2: Attack of the Clones            <data.table>
## 3: Return of the Jedi              <data.table>
## 4: Revenge of the Sith             <data.table>
## 5: The Empire Strikes Back         <data.table>
## 6: The Force Awakens               <data.table>
## 7: The Phantom Menace              <data.table>
```

Joining Revenue with Nested

```
# join nested and revenue on films variable
nested <- nested[revenue, on = "films"]
nested[]
```

```
##     films                data              revenue
##     <char>               <list>              <num>
## 1: A New Hope        <data.table>   775.4
## 2: The Empire ...    <data.table>   538.4
## 3: Return of ...     <data.table>   475.1
## 4: The Phantom ...   <data.table>  1027.0
## ...
```

After some analyses...

(Get counts of female characters per film; Could use dtplyr here instead)

```
# Get counts for each film from the data column
films[, counts := purrr::map(data, ~count(.x, gender))]
films[]
```

Now unnest the counts column (it is a list of tibbles)

```
films_unnest <- dt_unnest(films, counts)
films_unnest[]
```

The Unnested Data

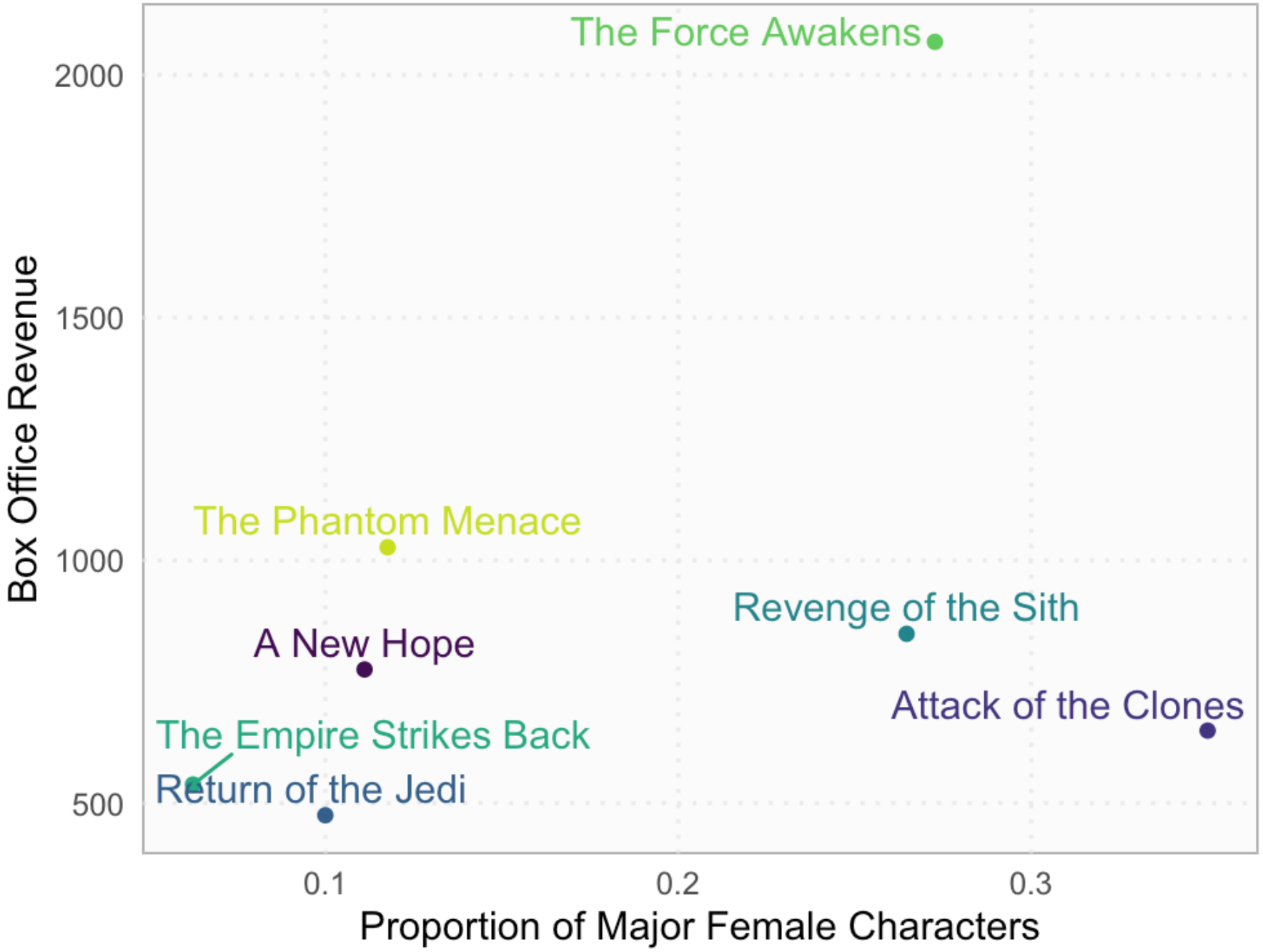
```
##     films                revenue gender           n
##     <char>               <num>   <char>           <int>
## 1: A New Hope            775.4   <NA>             3
## 2: A New Hope            775.4   female           2
## 3: A New Hope            775.4   hermaphrodite    1
## 4: A New Hope            775.4   male             12
## 5: The Empire ...        538.4   <NA>             2
## 6: The Empire ...        538.4   female           1
## 7: The Empire ...        538.4   male             12
## ...
```

From here, we can get the proportion of females to all characters and check the relationship between proportion and revenue

The

```
##
##                                                    int>
## 1
## 2
## 3
## 4                                                  2
## 5
## 6
## 7                                                  2
## .
```

From he
the relat



Cool, cool...

But what about performance?!

Nesting

```
#> # A tibble: 2 x 3
#>   expression   median mem_alloc
#>   <chr>      <bch:tm> <bch:byt>
#> 1 dt_nest      3.16ms    2.88MB
#> 2 group_nest   4.79ms    2.54MB
```

Unnesting

```
#> # A tibble: 2 x 3
#>   expression   median mem_alloc
#>   <chr>      <bch:tm> <bch:byt>
#> 1 dt_unnest     4.7ms    5.49MB
#> 2 unnest       12.6ms    8.47MB
```

Links and Such

Tyson S. Barrett

 tyson.barrett@usu.edu

 tysonbarrett.com

 @healthandstats

 github.com/tysonstanley

 Slides at tysonbarrett.com/teaching

How does `dt_hoist()` work?

the list-column with vectors in it

```
dt[,
   unlist(var, recursive = FALSE)),
   by = id]
```

The basic structure

How does `dt_nest()` work?

.SD is a placeholder for all variables, excluding the keyby variable(s)

The basic structure

```
dt[,
   list(list(.SD)),
   keyby = id]
```

keyby sorts the data using data.table's super fast sorting

the variable to nest by

An Example!

How does `dt_unnest()` work?

the <other variables> are all the other variables you want to hold on to

```
dt[,
   list(<other variables>,
        rbindlist(nested_col))]
```

rbindlist is data.table's super fast row binder